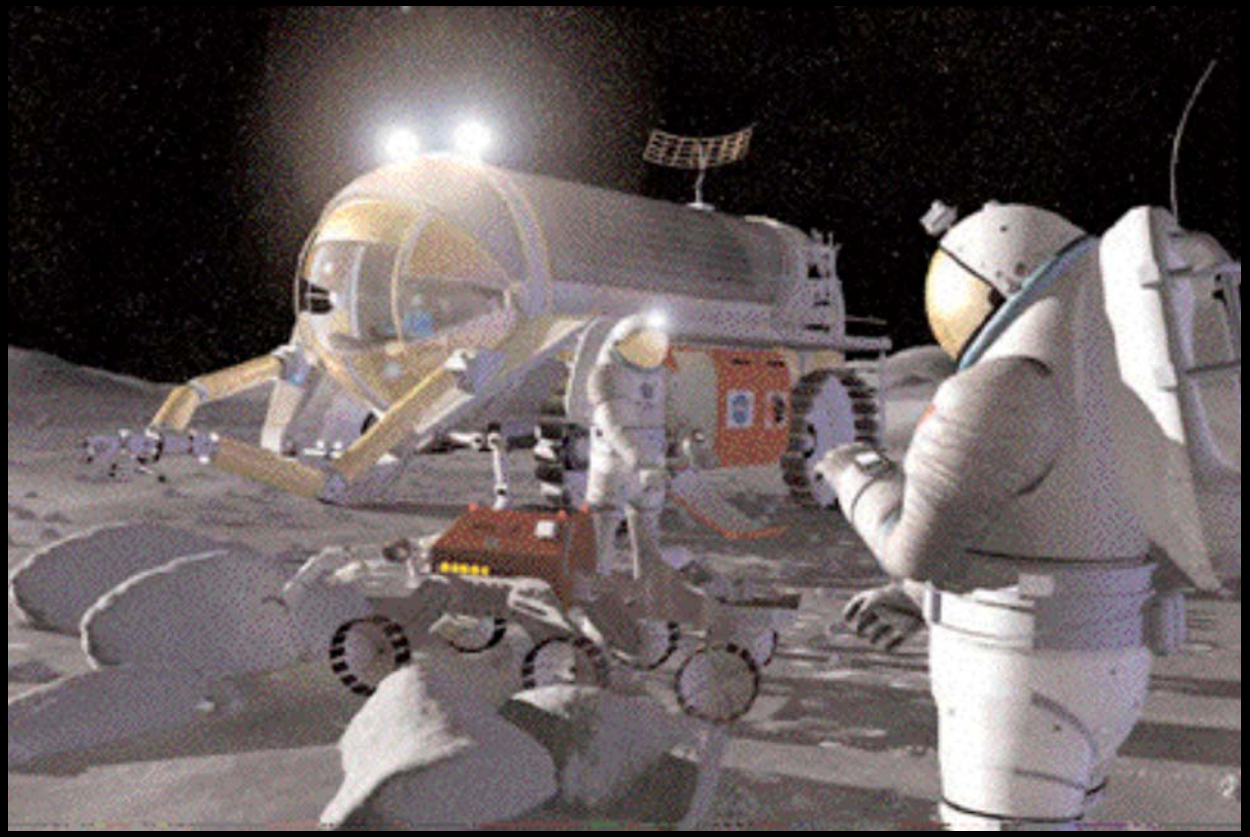
Trusted Autonomy for Spaceflight Systems



Michael Freed
NASA Ames Research Center
MS 262-4 Moffett Field, CA 94035
michael.a.freed@nasa.gov

Pete Bonasso	NASA JSC / TracLabs
Michel Ingham	JPL
David Kortenkamp	NASA JSC / TracLabs
Barney Pell	NASA Ames
John Penix	NASA Ames

Advanced automation in recent space missions

Deep Space 1
(1998 - 2001)

Remote Agent Experiment
- Planner/scheduler
- Smart Executive
- MIR

Earth Observing 1
(2000 – present)

Autonomous Sciencecraft Experiment
- CASPER planner
- SCL

Mars Exploration Rovers
(2003-present)

MAPGEN
- Europa planner

Outline

1. Advanced automation
2. Challenges and progress
   - Software reliability
   - Model reliability
   - Supporting gradual adoption
   - Achieving system maturity
3. Case study: advanced life support

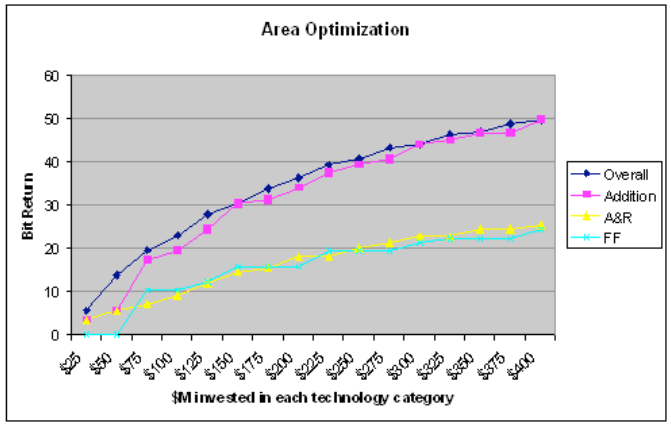
Advanced Automation

- NASA spacecraft have always used automation
- Advanced automation:
  - Goal-based commanding
  - Use of “intelligent” algorithms – based on theories of correct/optimal reasoning
    - Planning / scheduling
    - Adaptive execution
    - Failure, diagnosis, isolation, repair

Advanced Automation

Expected Benefits

- Reduced costs — see START project (Weisbin, JPL)
- Increased reliability — e.g. enhanced FDIR
- Increased productivity — MAPGEN 20%, Clarissa 100%, SCIP 300%
- Mission enablement — e.g. out of comm operations



ROI for tech investment options
http://start1.jpl.nasa.gov/index.cfm)

MAPGEN for MER sequencing

Dr. Michael Freed, NASA Ames Research Center

Trust and Adoption

Comparison to Aircraft Flight Automation

Gradual adoption the norm in most industries

Similarities

- Diverse roles (GNC to in-flight entertainment)
- System-of-systems level impact
- Handles many contingencies in moderately predictable environment
- Benefits in crew reduction, reliability, operational efficiency,…

Differences highlight challenges for spaceflight

- 1000s of flights/day facilitates gradual refinement and adoption
- Forgiving environment: problems unlikely to cause significant loss

We need faster progress in less forgiving conditions
We need to do it better

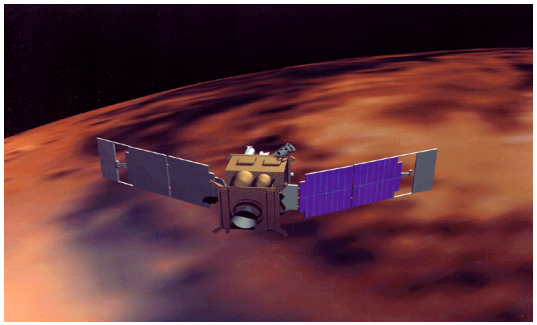
Challenges and Progress

Technology Risk Factors

- Software verification and validation
- Model validity
- Within-mission deployment pace
- Technology maturity

…ignoring other cost factors such as schedule risk and R&D cost

Software risk



Mars Climate Orbiter (Sept. 1999)

Mission: interplanetary weather satellite
Fate: No signal received after orbit insertion
Cause: failure to convert Imperial to metric units

Mars Polar Lander (Dec. 1999)

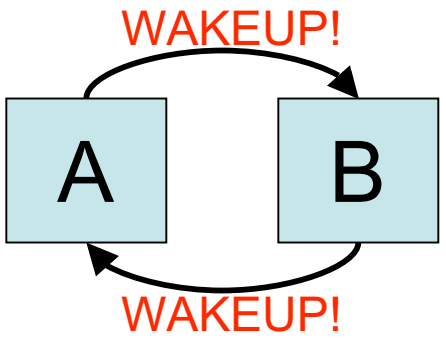
Mission: Dig for ice at Mars South Pole
Fate: No signal following initial descent
Cause: leg sensor noise led to premature engine off

Ariane 5 (June 1996)

Mission: deliver $500M payload to orbit
Fate: Veered off course and destroyed
Cause: Unhandled floating point exception

V&V of advanced automation software

Deadlock on DS-1 RAX

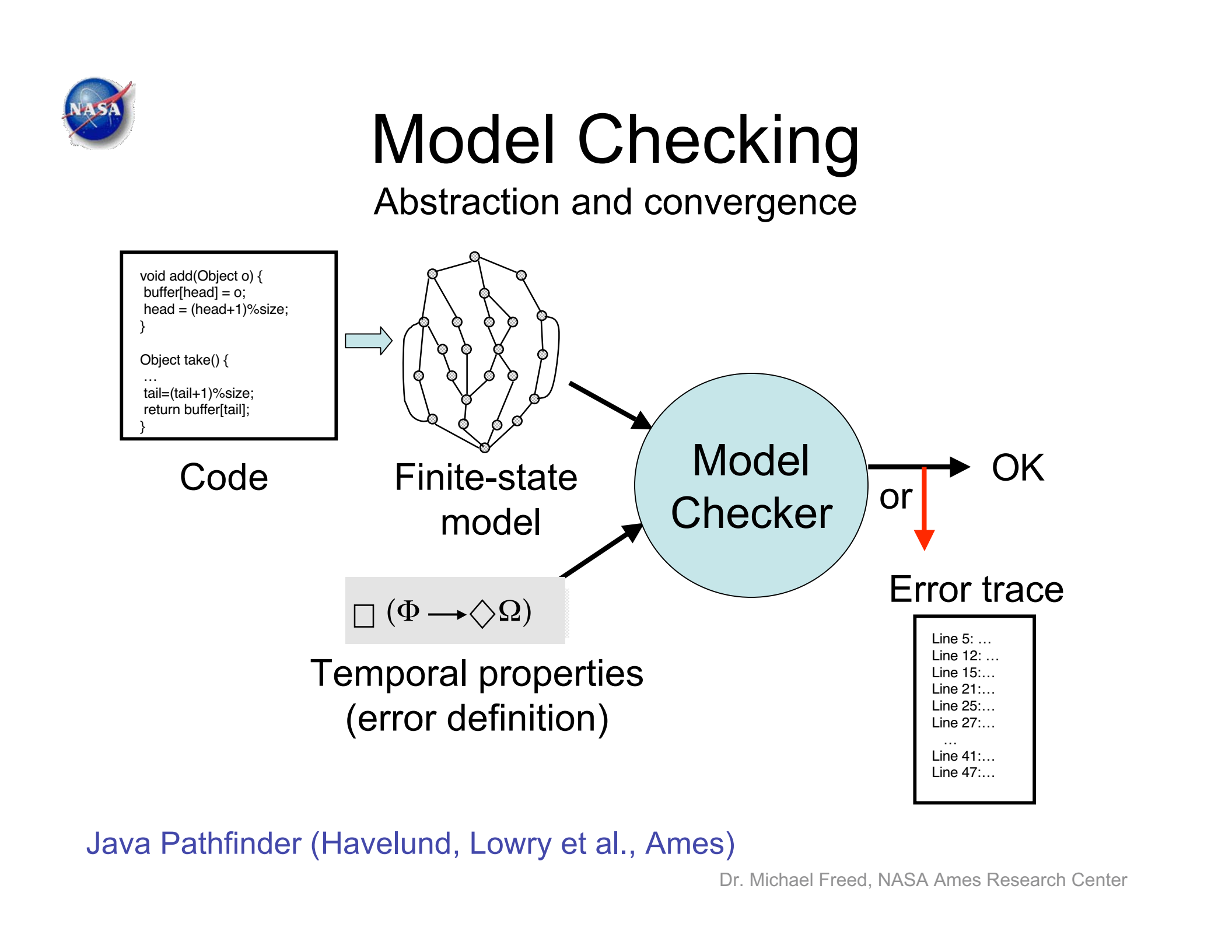



```
1. Go until done
2. Send wakeup
3. Put self to sleep
```

- Preemptive interrupt of A between steps 2, 3
- Process B starts, completes, sleeps
- Process A resumes, ignores wakeup since already running, then does step 3

* Now both A and B are asleep!

- Not caught in 800 hours of high-fidelity testing
- Concurrency errors (deadlocks, livelocks, race cond's,…) especially significant for advanced automation


Model Checking
Abstraction and convergence
void add(Object o) {
buffer[head] = o;
head = (head+1)%size;
}
Object take() {
…
tail=(tail+1)%size;
return buffer[tail];
}
Code
Finite-state model
Model Checker
or
OK
Error trace
Line 5: …
Line 12: …
Line 15:…
Line 21:…
Line 25:…
Line 27:…
…
Line 41:…
Line 47:…
□ (Φ ⟶ ◇Ω)
Temporal properties (error definition)
Java Pathfinder (Havelund, Lowry et al., Ames)
Dr. Michael Freed, NASA Ames Research Center

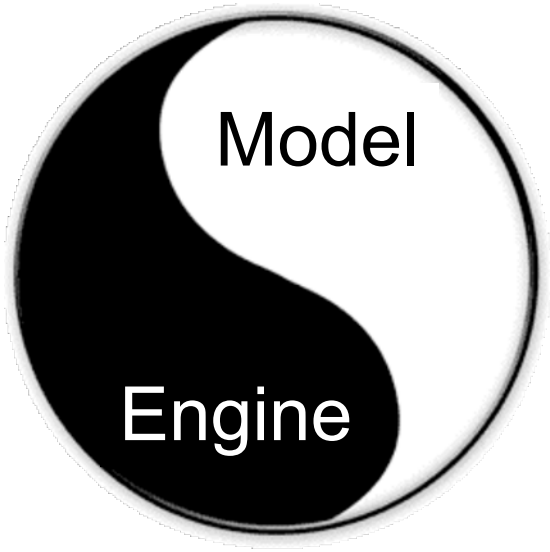
Analytical Software Verification Techniques

- Advanced Static Analysis
  - Analysis of static structure of code
  - Scale well, many false positives
- Model Checking
  - Exhaustive exploration of software states
  - General, good confidence, limited by # of states
- Runtime Verification
  - Explore one execution trace of a running program
  - Scale well, find many errors, but not all the errors

Models in Advanced Automation Software

Reasoning "Engine"
+ Model
Advanced Automation

Planner
Scheduler
Executive
FDIR



Physical structure
(connected tank-1 pipe-2)

Causal structure
(open valve-1)
⇒(connected pipe-2 tank-2)

"Physics"
(connected ?a ?b)
&(connected ?b ?c)
⇒(connected ?a ?c)

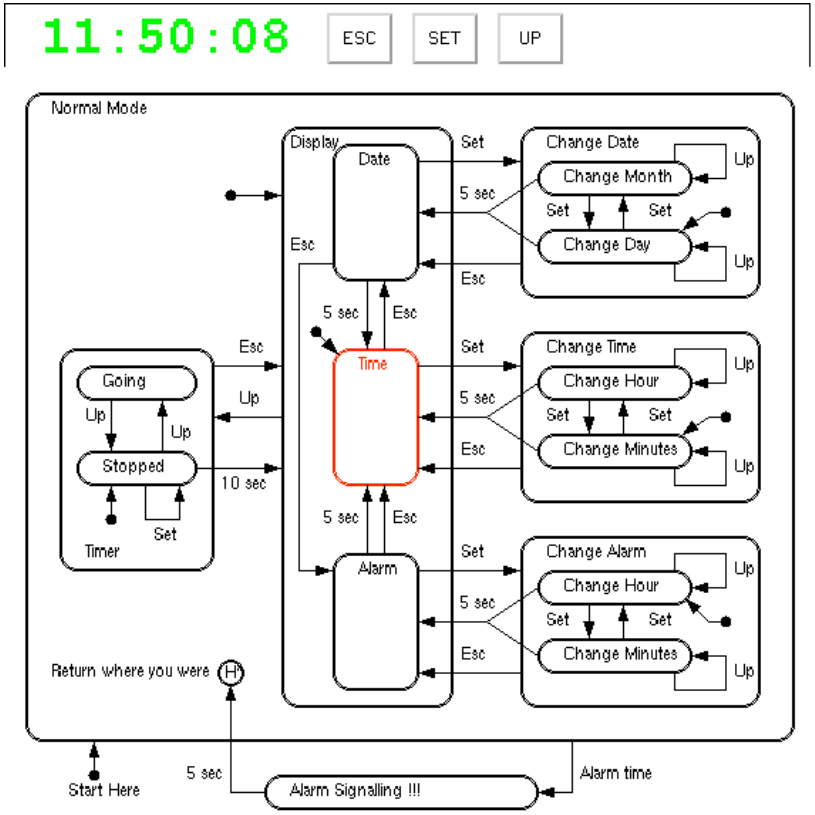
Policies
□ (open valve-1)
⇒ ~(open valve-2)

Models as a Source of Risk

- Model/engine separation is key source of capability for advanced automation
- Impact on risk is mixture of +'s and –'s
  - \+ Less error-prone software modification,…
  - \- Model inaccuracies lead to incorrect behavior
    1. Term ambiguity
    2. Underconstrained goals ("Genie's Wish Problem")
    3. Loss of context

Mitigating Model Risk

- In principle, engineers should be able to reliably create/ critique
- In practice, exotic formal notation requires AI specialists to translate
- Addressing the problem
  - ➢ Use more readable notations
  - ➢ Develop knowledge elicitation and update methodologies
  - ➢ Simulation-based review



State Chart graphical notation

Supporting Gradual Adoption

How much automation?

- At early phase, minimal automation may be safest
- Want to minimize required leaps of faith
- So, better to allow degree of automation to be adjusted than to fix during design

  (though see Proud, Hart, Mrozinski at JSC)

Classic Automation Levels

1. The computer offers no assistance, human must do it all
2. The computer offers a complete set of action alternatives and
3. Narrows the selection down to a few,or
4. Suggests one, and
5. Executes that suggestion if the human approves, or
6. Allows the human a restricted time to veto before automatic execution, or
7. Executes automatically, then necessarily informs the human, or
8. Informs human after execution afterwards only if asked, or
9. Informs human after execution if computer decides to
10. The computer decides everything, and acts autonomously, ignoring human

(Sheridan 1992)

Classic treatment of “degrees of automation” too crude to guide development of Exploration systems

Flexibly Adjustable Automation

Need ability to independently adjust specific decision processes -- e.g. when next to purge nitrifier line

- Command complexity (goal-based vs. action-based)
- The resources (including time) consumed by its operation
- Circumstances in which it will override/allow manual control
- Circumstances in which it will request decision from user
- Need coarse-grained control also – e.g. ability to turn off automation in subsystem

Achieving Flexible Adjustability

Not an independent technology, but an advance in systems engineering practice.

1. Identify and instrument all automation decision points
2. Human interfaces and human interaction design must
   - support intermittently reacquiring situation awareness
   - facilitate most common or speed-critical adjustments

#1 Lesson from DS-1 (RAX), ALS & other experiences:
Early integration of automation team with other teams critical. Building automation after other systems does not work well.

System maturity

No one wants to use new automation technology in a critical role. Test deployments of Exploration automation should start long before anticipated use.

Advantages of long maturation strategy:

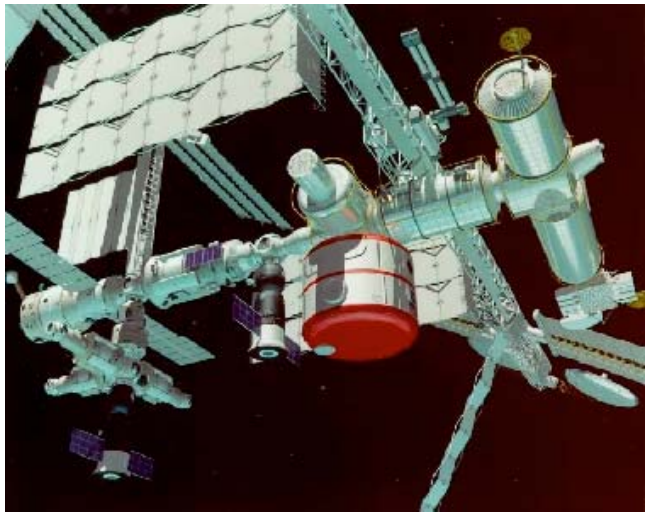
- Time to shake out bugs, improve performance
- Opportunity to analyze performance and failure modes
- Time to compile documentation, consult with users
- Increases number of people familiar with technology, source of technical guidance and staffing

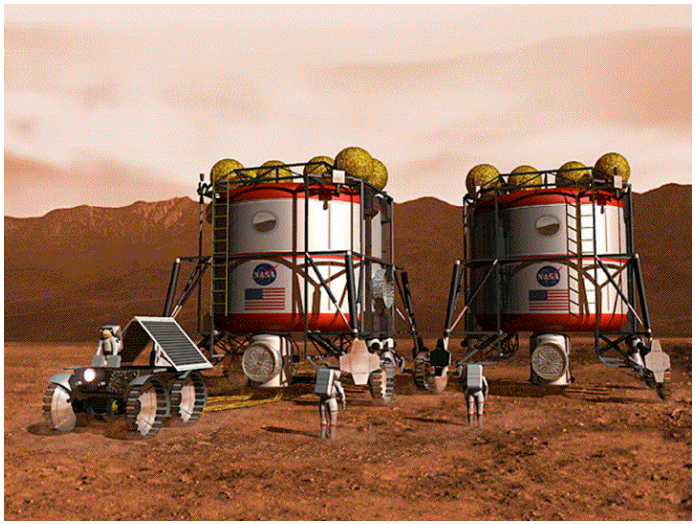
Case Study: JSC Advanced Life Support Tests

7 years and counting

Advanced (regenerative/closed) life support

- Many complex interactions between subsystems
- Challenging to control / optimize
- Not suitable for human vigilant monitoring
- Critical to have trustworthy automation

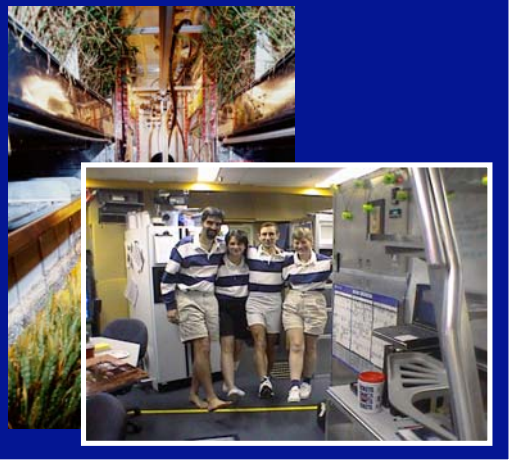




Case Study: Advanced Life Support Tests

Plant Growth

Air Revitalization



Water Recovery

NASA JSC ALS Testbed: 3 major systems + crew, many subsystems

3T advanced automation architecture used for control

- ran 24/7 with 200+ sensors/actuators (AWRS)
- complex research system with uptime >98%

ALS Summary

Life support engineers shifted from vigilant monitoring to remote intermittent supervision. (Paradigm shift)

Lessons learned (more documented elsewhere)

1. Trust emerges gradually: “flight following” to vigilant monitoring to supervision over several years
2. Once basic system (“engine”) trusted, trust emerges faster: 30 days for AWRS
3. Designing automation and hardware concurrently key to good systems engineering and eventual trust
4. Crucial to be able to selectively adjust automation role for shakeout, maintenance, upgrades
5. Achieving needed adjustability depends on understanding human-automation interaction

Conclusion

- Big challenges; big rewards for succeeding
- New developments, particularly by people at NASA, are fast expanding the capabilities automation offers for given level of risk
- Long-term needs (spiral 3) require early investment
- Long-term deployments are key to addressing trust challenges